Title	The Membrane Systems Language Class (Mathematical Foundation of Algorithms and Computer Science)
Author(s)	ALHAZOV, Artiom; CIUBOTARU, Constantin; ROGOZHIN, Yurii; IVANOV, Sergiu
Citation	数理解析研究所講究録 (2010), 1691: 44-50
Issue Date	2010-06
URL	http://hdl.handle.net/2433/141574
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録
第1691巻 2010年 44-50

The Membrane Systems Language Class*

Artiom ALHAZOV[1,2], Constantin CIUBOTARU[1], Yurii ROGOZHIN[1], Sergiu IVANOV[1,3]
[1] Institute of Mathematics and Computer Science, Academy of Sciences of Moldova,
Academiei 5, Chişinău MD-2028 Moldova
E-mail: {artiom,chebotar,rogozhin,sivanov}@math.md
[2] IEC, Department of Information Engineering, Graduate School of Engineering
Hiroshima University, Higashi-Hiroshima 739-8527 Japan
[3] Technical University of Moldova
Ştefan cel Mare 168, Chişinău MD-2004 Moldova

Abstract This paper introduces the class of languages generated by the transitional model of membrane systems without cooperation and without additional ingredients. The fundamental nature of these basic systems allows us to define this class of languages it in terms of derivation trees of context-free grammars. We compare this class to the well-known language classes and discuss its properties.

1 Introduction

Membrane computing is a theoretical framework of parallel distributed multiset processing. It has been introduced by Gheorghe Păun in 1998, and remains an active research area, see [6] for the comprehensive bibliography and [3],[4] for a systematic survey.

The configurations of membrane systems (with symbol objects) consist of multisets over a finite alphabet, distributed across a tree structure. Therefore, even such a simple structure as a word (i.e., a sequence of symbols) is not explicitly present in the system. To speak of languages as sets of words, one first needs to represent them in membrane systems, and there are a few ways to do it.

The first way is to represent words by string objects. Rather many papers take this approach, see Chapter 7 of [4], but only few consider parallel operations on words. Moreover, a tuple of sets or multisets of words is already a quite complicated structure. The third drawback is that it is very difficult to define an elegant way of interactions between strings. Examples are polarizations and splicing, but these are difficult to use in applications. Here we deal with the symbol objects.

The second way is to represent a word by a single symbol object, or by a few objects of the form (letter,position) as in, e.g., [1]. One can only speak about finite languages in this way.

The third way is to represent positions of the letters in a word by nested membranes. The corresponding letters can be encoded by objects in the associated regions, membrane types or membrane labels. Such a representation requires sophisticated types of rules, [2].

The fourth way is to consider letters as digits and then view words as numbers, or use some other encoding of words into numbers or multisets. Clearly, the concept of words is no longer direct, and implementing basic word operations in this way requires a lot of number processing.

The fifth way is to work with multisets, and regard the order of sending the objects in the environment as their order in the output word. The class of languages of our interest is the class generated by systems with parallel applications of non-cooperative rules rewriting objects and sending them between the regions. Surprisingly, this class has not received enough attention. Almost all known characterizations and bounds for generative power of different membrane systems with various ingredients and descriptional complexity bounds are expressed in terms of REG, MAT, $ET0L$ and RE, their length sets and Parikh sets (and much less often in terms of FIN, other subregular classes, CF or CS). The membrane systems language class lies between regular and context-sensitive classes, being incomparable with well-studied intermediate ones.

*Artiom Alhazov acknowleges the support of the Japan Society for the Promotion of Science and the Grant-in-Aid for Scientific Research, project 20-08364. All authors acknowledge the support by the Science and Technology Center in Ukraine, project 4032.

2 Definitions

We start with some formal language preliminaries. Consider a finite set V. The set of all words over V is denoted by V^*, the concatenation operation is denoted by $\bullet$ and the empty word is denoted by λ. Any set $L \subseteq V^*$ is called a language. For a word $w \in V^*$ and a symbol $a \in V$, the number of occurrences of a in w is written as $|w|_a$. The permutations of a word $w \in V^*$ are $\texttt{Perm}(w) = \{x \in V^* \mid |x|_a = |w|_a \forall a \in V\}$. We denote the set of all permutations of the words in L by $\texttt{Perm}(L)$, and we extend this notation to classes of languages. We use FIN, REG, LIN, CF, MAT, CS, RE to denote finite, regular, linear, context-free, matrix, context-sensitive and recursively enumerable families of languages, respectively. The family of languages generated by extended (tabled) interactionless L systems is denoted by $E(T)0L$. For more preliminaries see [5].

Throughout this paper we use string notation to denote the multisets. When speaking about membrane systems, the order in which symbols are written is irrelevant.

2.1 Transitional P systems

A membrane system is defined by a tuple $\Pi = (O, \mu, w_1, \cdots, w_m, R_1, \cdots, R_m, i_0)$, where O is a finite set of objects, μ is a hierarchical structure of m membranes, bijectively labeled by $1, \cdots, m$; the interior of each membrane defines a region; the environment is referred to as region 0, w_i is the initial multiset in region i, $1 \leq i \leq m$, R_i is the set of rules of region i, $1 \leq i \leq m$, and $i_0 = 0$ is the output region.

The rules of a membrane systems have the form $u \to v$, where $u \in O^+$, $v \in (O \times Tar)^*$. The target indications from $Tar = \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$ are written as a subscript, and target $here$ is omitted. In case of non-cooperative rules, $u \in O$.

The rules are applied in maximally parallel way: no further rule should be applicable to the idle objects. In case of non-cooperative systems, the concept of maximal parallelism is the same as evolution in L systems: all objects evolve by the associated rules in the corresponding regions (except objects a in regions i such that R_i does not contain any rule $a \to u$, but these objects do not contribute to the result). The choice of rules is non-deterministic.

A sequence of transitions is called a computation. The computation halts when such a configuration is reached that no rules are applicable. The result of a (halting) computation is the *sequence* of objects sent to the environment (all the permutations of the symbols sent out in the same time are considered). The language $L(\Pi)$ generated by a P system Π is the union of the results of all computations. The class of languages generated by non-cooperative transitional P systems with at most m membranes is denoted by $LOP_m(ncoo, tar)$. If the number of membranes is not bounded, m is replaced by $*$ or omitted. If the target indications of the form in_j are not used, tar is replaced by out.

Example 1 *To illustrate the concept of generating languages, consider the following P system:* $\Pi = (\{a, b, c\}, [_1 \ ]_1, a^2, \{a \to \lambda, a \to a\, b_{out} c^2_{out}\}, 0)$. Each of the two symbols a has a non-deterministic choice whether to be erased or to reproduce itself while sending a copy of b and two copies of c into the environment. Therefore, the contents of region 1 can remain a^2 for an arbitrary number $m \geq 0$ of steps, and after that at least one copy of a is erased. The other copy of a can reproduce itself for another $n \geq 0$ steps before being erased. Each of the first m steps, two copies of b and four copies of c are sent out, while in each of the next n steps, only one copy of b and two copies of c are ejected. Therefore, $L(\Pi) = (\texttt{Perm}(bccbcc))^*(\texttt{Perm}(bcc))^*$.

3 Time yield of CF grammars

Consider a grammar $G = (N, T, S, P)$ and $A \in N$. We denote by G_A the grammar (N, T, A, P) obtained by considering A as axiom in G.

A derivation tree in a context-free grammar is always a rooted tree with leaves labeled by terminals and all other nodes labeled by non-terminals. Rules of the form $A \to \lambda$ cause a problem, which can be solved by allowing to also label leaves by λ, or by transformation of the corresponding grammar. Note: we only consider finite derivation trees. We define n-th level yield of a derivation tree τ:

We define $\texttt{yield}_0(\tau) = a$ if τ has a single node labeled by $a \in T$, and $\texttt{yield}_0(\tau) = \lambda$ otherwise.

Let k be the number of children nodes of the root of τ, and $\tau_1, \cdots, \tau_k$ be the subtrees of τ with these children as roots. We define $\texttt{yield}_{n+1}(\tau) = \texttt{yield}_n(\tau_1) \bullet \texttt{yield}_n(\tau_2) \bullet \cdots \bullet \texttt{yield}_n(\tau_k)$.

We now define the time yield L_t of a context-free grammar derivation tree τ, as the usual yield except the order of terminals is vertical from root instead of left-to-right, and the order of terminals at the same distance from root is arbitrary. We use $\prod$ to denote concatenation in the following definition:

$$L_t(\tau) = \prod_{n=0}^{\texttt{height}(\tau)} (\texttt{Perm}(\texttt{yield}_n(\tau))).$$

The time yield $L_t(G)$ of a grammar G is the union of time yields of all its derivation trees. The corresponding class of languages is $L_t(CF) = \{L_t(G) \mid G \text{ is a CF grammar}\}$.

Example 2 *Consider a grammar* $G_1 = (\{S, A, B, C\}, \{a, b, c\}, S, P)$, $P = \{S \to SABC, S \to ABC, A \to A, B \to B, C \to C, A \to a, B \to b, C \to c\}$. *We now show that* $L_t(G_1) = \{w \in \{a, b, c\}^* \mid |w|_a = |w|_b = |w|_c > 0\} = L$. *Indeed, all derivations of* A *are of the form* $A \Rightarrow^* A \Rightarrow a$. *Likewise, symbols* B, C *are also trivially rewritten an arbitrary number of times and then changes into a corresponding terminal. Hence,* $L_t(G_{1A}) = \{a\}$, $L_t(G_{1B}) = \{b\}$, $L_t(G_{1C}) = \{c\}$. *For inclusion* $L_t(G) \subseteq L$ *it suffices to note that* S *always generates the same number of symbols* A, B, C.

The converse inclusion follows from the following simulation: given a word $w \in L$, *generate* $|w|/3$ *copies of* A, B, C, *and then apply their trivial rewriting such that the timing when the terminal symbols appear corresponds to their order in* w.

Corollary 1 $L_t(CF) \not\subseteq CF$.

4 Membrane class via derivation trees of CF grammars

We first show that for every membrane system without cooperation, there is an equivalent system from the same class with one membrane.

Lemma 1 $LOP(ncoo, tar) = LOP_1(ncoo, out)$.

Proof. Consider an arbitrary transitional membrane system Π (without cooperation and without additional ingredients). The known technique of flattening the structure consists of transforming Π as follows. Object a in region associated to membrane i is transformed into object (a, i) in the region associated to the single membrane. The alphabet, initial configuration and rules are transformed accordingly. Clearly, the configurations of the old system and the new system are isomorphic, and the output in the environment is the same. □

Theorem 1 $L_t(CF) = LOP(ncoo, tar)$.

Proof. By Lemma 1, the statement is equivalent to $L_t(CF) = LOP_1(ncoo, out)$. Consider a P system $\Pi = (O, [_1\ ]_1, w, R, 0)$. We construct a context-free grammar $G = (O' \cup \{S\}, O, S, P \cup \{S \to w\})$, where S is a new symbol, $'$ is a morphism from O into new symbols and $P = \{a' \to u'v \mid (a \to u\ v_{out}) \in R, a \in O, u, v \in O^*\} \cup \{a' \to \lambda \mid \neg\exists (a \to u\ v_{out}) \in R\}$. Here v_{out} are those symbols on the right side of the rule in R which are sent out, and u are the remaining right-side symbols.

The computations of Π are identical to parallel derivations in G, except the following points. First, unlike G, Π does not keep track of the left-to-right order of symbols. This does not otherwise influence the derivation (the rules are context-free) or the result (the order of nonterminals produced in the same step is arbitrary, and the timing is preserved). Second, the initial configuration of Π is produced from the axiom of G in one additional step. Third, the objects of Π that cannot evolve are erased in G, since they do not contribute to the result.

It this way $L_t(CF) \supseteq LOP(ncoo, tar)$. To prove the converse inclusion, consider an arbitrary context-free grammar $G = (N, T, S, P)$. We construct a P system $\Pi = (N \cup T, [_1\ ]_1, S, R, 0)$, where $R = \{a \to h(u) \mid (a \to u) \in R\}$, where h is a morphism defined by $h(a) = a$, $a \in N$ and $h(a) = a_{out}$, $a \in T$. The computations in Π correspond to parallel derivations in G, and the order of producing

terminal symbols in G corresponds to the order of sending them to the environment by Π. □

We now present a few normal forms.

Lemma 2 *(First normal form) For a context-free grammar G there exists a context-free grammar G' such that $L_t(G) = L_t(G')$ and*

- *the axiom of G' does not appear in the right side of any rule, and*
- *if the left side is not the axiom in G', then the right side is not empty.*

Proof. The technique is essentially the same as removing λ-productions in classical theory of context-free grammars. Let $G = (N, T, S, P)$. First, introduce the new axiom S' and add a rule $S' \to S$. Compute the set $E \subseteq N$ of non-terminals that can derive λ by closure of $(A \to \lambda) \longrightarrow (A \in E)$ and

$$\left.\begin{array}{l}(A \to A_1 \cdots A_k),\\ (A_1, \cdots, A_k \in E)\end{array}\right\} \longrightarrow (A \in E).$$

Then replace productions $A \to u$ by $A \to h(u)$, where $h(a) = \{a, \lambda\}$ if $a \in E$ and $h(a) = a$ if $a \in N \cup T \setminus E$. Finally, remove λ-productions for all non-terminals except the axiom (this preserves not only the generated terminals, but also the order in which they are generated). □

The First normal form shows that erasing can be limited to the axiom.

Lemma 3 *(Binary normal form) For a context-free grammar G there is a context-free grammar G' such that $L_t(G) = L_t(G')$ and*

- *G' is in the First normal form,*
- *the right side of productions is at most 2.*

Proof. The only concern in splitting the longer productions of $G = (N, T, S, P)$ in shorter ones is to preserve the order in which non-terminals are produced. The number $n = \lceil \log_2 (\max_{(A \to u) \in P} |u|) \rceil$ is the number of steps sufficient to implement all productions of G by at most binary productions. Each production $p : A \to A_1 \cdots A_k$, $k \leq 2^n$, is replaced by productions implementing a binary tree, rooted in A with new symbols in intermediate nodes, and leaves labeled $A_1, \cdots, A_k$ at depth n. □

The Binary normal form shows that productions with right side > 2 are not needed.

Lemma 4 *(Third normal form) For a context-free grammar G there exists a context-free grammar G' such that $L_t(G) = L_t(G')$ and*

- *G' is in the Binary normal form,*
- *$G' = (N, T, S, P')$; $\forall A \in N$ is reachable,*
- *either $G' = (\{S\}, T, S, \{S \to S\})$, or $G' = (N, T, S, P')$ and $\forall A \in N$, $L_t(G'_A) \neq \emptyset$.*

Proof. Consider a CF grammar in the Binary normal form. First, compute the set $D \subseteq N$ of productive non-terminals as closure of $(A \to u)$, $(u \in T^*) \longrightarrow (A \in D)$ and

$$\left.\begin{array}{l}(A \to A_1 \cdots A_k),\\ (A_1, \cdots, A_k \in D)\end{array}\right\} \longrightarrow (A \in D).$$

Remove all non-terminals that are not productive from N, and all productions containing them. If the axiom was also removed, then $L_t(G) = \emptyset$, so we can take $G' = (\{S\}, T, S, \{S \to S\})$. Otherwise, compute the set $R \subseteq N$ of reachable non-terminals as closure of $(S \in R)$ and

$$\left.\begin{array}{l}(A \in R),\\ (A \to A_1 \cdots A_k)\end{array}\right\} \longrightarrow (A_1, \cdots, A_k \in R).$$

Remove all non-terminals that are not reachable from N, and all productions containing them. All transformations preserve the generated terminals and the order of their production, as well as the Binary normal form. □

The Third normal form shows that non-ending derivations are only needed for $\emptyset$.

5 Comparing the class

Theorem 2 $LOP(ncoo, tar) \supseteq REG$.

Proof. Any regular language is accepted by some complete finite automaton $M = (Q, \Sigma, q_0, F, \delta)$. We construct a context-free grammar $G = (Q, \Sigma, q_0, P)$, where $P = \delta \cup$

$\{q \to \lambda \mid q \in F\}$. The order of symbols accepted by M corresponds to the order of symbols generated by G, and the derivation only finishes when the final state is reached. □

Theorem 3 $LOP(ncoo, tar) \subseteq CS$.

Proof. Consider a context-free grammar $G = (N, T, S, P)$ in the First normal form. We construct a grammar $G' = (N \cup \{\#_1, L, R, F, \#_2\}, T, S', P')$, $P' = \{S' \to \#_1 LS\#_2, L\#_2 \to R\#_2, \#_1 R \to \#_1 L, \#_1 R \to F, F\#_2 \to \lambda\} \cup \{LA \to uL \mid (A \to u) \in P\} \cup \{La \to aL, Fa \to aF \mid a \in T\} \cup \{aR \to Ra \mid a \in N \cup T\}$. The symbols $\#_1, \#_2$ mark the edges; symbol L applies productions P to all non-terminals, left-to-right, skipping the terminals. At the end marker, symbol L changes into R and returns to the beginning marker, where it either changes back to L to iterate, or to F to check if the derivation is finished.

Hence, $L(G') = L_t(G)$. The length of sentential forms in any derivation (of a word with n symbols in G') is at most $n+3$, because the only shortening productions are the ones removing $\#_1, \#_2$ and F, and each is applied once. □

To show that the membrane systems language class does not contain the class of linear languages, we first define the notions of unbounded yield and unbounded time.

Definition 1 *Take a grammar $G = (N, T, S, P)$. We say that $A \in N$ has an unbounded yield if $L_t(G_A)$ is an infinite language, i.e., there is no upper bound on the length of words generated from A.*

Clearly, $L_t(G_A)$ is infinite if and only if $L(G_A)$ is infinite; decidability of this property is well-known from the theory of CF grammars.

Definition 2 *In a grammar $G = (N, T, S, P)$, we say that $A \in N$ has unbounded time if the set of all derivation trees in G_A is infinite. i.e., there is no upper bound on the number of parallel steps of terminated derivations in G_A.*

Clearly, A has unbounded time if $L(G_A) \neq \emptyset$ and $A \Rightarrow^+ A$. Decidability of this property is well-known from the CF grammar theory.

Lemma 5 *Let $G = (N, T, P, S)$ be a context-free grammar in the Third normal form. If for every rule $(A \to BC) \in P$, symbol B does not have unbounded time, than $L_t(G) \in REG$.*

Proof. Assume the premise of the lemma. Let F be the set of the first symbols in the right sides of all binary productions. There exists a maximum m of time bounds for the symbols in F. For every such symbol $B \in F$ there also exists a finite set $t(B)$ of derivation trees in G_B. Let $t = \{\emptyset\} \cup \bigcup_{B \in F} t(B)$ be the set of all such derivation trees, also including the empty tree. Recall that t is finite.

We perform the following transformation of the grammar: we introduce non-terminals of the form $A[\tau_1, \cdots, \tau_{m-1}]$, $A \in N \cup \emptyset$, $\tau_i \in t$, $1 \leq i \leq m-1$. The new axiom is $S[\emptyset, \cdots, \emptyset]$. Every binary production $A \to BC$ is replaced by productions $A[\tau_1, \cdots, \tau_{m-1}] \to \mathtt{yield}_0(\tau)\mathtt{yield}_1(\tau_1) \cdots \mathtt{yield}_{m-1}(\tau_{m-1})C[\tau, \tau_1, \cdots, \tau_{m-2}]$ for all $\tau \in t(B)$. Accordingly, productions $A \to C$, $C \in N$ are replaced by productions $A[\tau_1, \cdots, \tau_{m-1}] \to \mathtt{yield}_1(\tau_1) \cdots \mathtt{yield}_{m-1}(\tau_{m-1})C[\emptyset, \tau_1, \cdots, \tau_{m-2}]$, and productions $A \to a$, $a \in T$ are replaced by $A[\tau_1, \cdots, \tau_{m-1}] \to a\ \mathtt{yield}_1(\tau_1) \cdots \mathtt{yield}_{m-1}(\tau_{m-1})\emptyset[\emptyset, \tau_1, \cdots, \tau_{m-2}]$. Finally, $\emptyset[\emptyset, \cdots, \emptyset] \to \lambda$ and

$$\emptyset[\tau_1, \cdots, \tau_{m-1}] \to \mathtt{yield}_1(\tau_1) \cdots \mathtt{yield}_{m-1}(\tau_{m-1})\emptyset[\emptyset, \tau_1, \cdots, \tau_{m-2}].$$

If the effect of one symbol is limited to m steps, then the choice of the corresponding derivation tree is memorized as an index in the other symbol, and needed terminals are produced in the right time. In total, m indexes suffice. The grammar is regular, since only one non-terminal is present. □

Lemma 6 $\{a^n b^n \mid n \geq 1\} \notin LOP(ncoo, tar)$.

Proof. Denote $\{a^n b^n \mid n \geq 1\}$ by L. Suppose $G = (N, T, S, P)$ is a context-free grammar in the Third normal form and $L_t(G) = L$. Clearly, there must be a rule $A \to BC$ or $A \to CB \in P$ such that both B and C have unbounded

time (by Lemma 5, since $L \notin REG$) and C has unbounded yield (since $L \notin FIN$).

Languages generated by any non-terminal from N must be scattered subwords of words from L, otherwise G would generate some language not in L. Thus, $L_t(G_B), L_t(G_C) \subseteq \{a^i b^j \mid i,j \geq 0\}$. Clearly, G_C must produce both symbols a and b. Indeed, since the language generated from C is infinite, substituting derivation trees for C with different numbers of one letter must preserve the balance of two letters. Consider two cases, depending on whether $L_t(G_B) \subseteq a^*$.

If B only produces symbols a, then consider the shortest derivation tree τ in G_C. Since B has unbounded time, some symbol a can be generated after the first letter b appears in τ, so $L_t(G) \not\subseteq L$, which is a contradiction.

Now assume B can produce a symbol b in some derivation tree τ in G_B. On one hand, a bounded number of letters a can be generated from B and C before the first letter b appears in τ; on the other hand, C has unbounded yield. Varying derivations under C we get an infinite subset of $L_t(G)$ with a bounded number of leading symbols a, so $L_t(G) \not\subseteq L$. □

Corollary 2 $LIN \not\subseteq LOP(ncoo, tar)$.

Lemma 7 *The class $LOP(ncoo, tar)$ is closed under permutations.*

Proof. In a given grammar $G = (N, T, S, P)$, replace the terminals a by non-terminals a_N throughout the description of G, and then add rules $a_N \to a_N$, $a_N \to a$ to P, $a \in T$. The terminals are generated in arbitrary order. □

Corollary 3 $\mathtt{Perm}(REG) \subseteq LOP(ncoo, tar)$.

Summarizing the position of the membrane system class w.r.t the Chomsky hierarchy,

Theorem 4 *$LOP(ncoo, tar)$ strictly contains REG and $\mathtt{Perm}(REG)$, is strictly contained in CS, and is incomparable with LIN and CF.*

The lower bound can be strengthened:

Theorem 5 $REG \bullet \mathtt{Perm}(REG) \subseteq LOP(ncoo, tar)$.

Proof. Consider the construction from the regularity theorem. Rewrite the symbol corresponding to the final state, into the axiom of the grammar generating the second regular language, to which the permutation technique is applied. □

Example 3 $L_2 \in LOP(ncoo, tar)$, $L_2 = \bigcup_{m,n \geq 1} (abc)^m \mathtt{Perm}((def)^n)$.

6 Closure properties

Following the permutations, we present a few other closure properties.

Lemma 8 *The class $LOP(ncoo, tar)$ is closed under erasing/renaming morphisms.*

Proof. Without restricting generality, we assume that the domain and range of a morphism h are disjoint. For a given grammar $G = (N, T, S, P)$, consider a transformation where the terminal symbols become non-terminals and the rules $a \to h(a)$, $a \in T$ are added to P. The new grammar generates $h(L_t(G))$. □

Corollary 4 $\{a^n b^n c^n \mid n \geq 1\} \notin LOP(ncoo, tar)$.

Proof. By contrary, reducing to Lemma 6. □

Corollary 5 *$LOP(ncoo, tar)$ is not closed under intersection with regular languages.*

Proof. By Example 2, $L = \{w \in T^* \mid |w|_a = |w|_b = |w|_c > 0\}$ belongs to the membrane systems language class. However, $L \cap a^*b^*c^* = \{a^n b^n c^n \mid n \geq 1\}$ does not, by Corollary 4. □

Theorem 6 *$LOP(ncoo, tar)$ is closed under union but not intersection or complement.*

Proof. The closure under union follows from adding a new axiom and productions of nondeterministic choice between multiple axioms. The class is not closed under intersection because it contains all regular languages (Theorem 2) and is not closed under intersection with them (Corollary 5). This class is not closed under complement, since intersection is the complement of union of complements. □

Lemma 9 $L \notin LOP(ncoo, tar)$, $L = \bigcup_{m,n\geq 1} \mathtt{Perm}((ab)^m)c^n$.

Proof. Suppose there exists a context-free grammar $G = (N, T, S, P)$ in the Third normal form such that $L_t(G) = L$. Clearly, there must be a rule $A \to BC$ or $A \to CB \in P$ such that both B and C have unbounded time (by Lemma 5, since $L \notin REG$) and C has unbounded yield (since $L \notin FIN$). By choosing as $A \to BC$ or $A \to CB$ the rule satisfying above requirements which is first applied in some derivation of G, we make sure that all three letters a, b, c appear in words of $L_t(G_A)$.

Languages generated by any non-terminal from N must be scattered subwords of words from L, otherwise G would generate some language not in L. Thus, $L_t(G_B), L_t(G_C) \subseteq \{a, b\}^*c^*$. Consider two cases, depending on whether $L_t(G_B) \subseteq \{a, b\}^*$.

If B only produces symbols a, b, then consider the shortest derivation tree τ in G_C. Since B has unbounded time, some symbol a or b can be generated after the first letter c appears in τ, so $L_t(G) \not\subseteq L$, a contradiction.

Now assume B can produce a symbol c in some derivation tree τ in G_B. On one hand, a bounded number of letters a, b can be generated from B and C before the first letter c appears in τ; on the other hand, C has unbounded yield. Varying derivations under C we obtain an infinite subset of $L_t(G)$ where the number of leading symbols a, b is bounded, so $L_t(G) \not\subseteq L$. □

Corollary 6 *$LOP(ncoo, tar)$ is not closed under concatenation or taking the mirror image.*

Proof. Since $\bigcup_{m\geq 1} \mathtt{Perm}((ab)^m) \in \mathtt{Perm}(REG) \subseteq LOP(ncoo, tar)$ by Corollary 3 and $c^+ \in REG \subseteq LOP(ncoo, tar)$ by Theorem 2, the first part of the statement follows from Lemma 9. Since $\bigcup_{m,n\geq 1} c^n\mathtt{Perm}((ab)^m) \in REG \bullet \mathtt{Perm}(REG) \subseteq LOP(ncoo, tar)$ by Theorem 5, the second part of the statement also follows from Lemma 9. □

7 Conclusions

We have reconsidered the class of languages generated by transitional P systems without cooperation and without additional control. It was shown that one membrane is enough, and a characterization of this class was given via derivation trees of context-free grammars. Next, three normal forms were given for the corresponding grammars. It was than shown that the membrane systems language class lies between $REG \bullet \mathtt{Perm}(REG)$ and context-sensitive languages, and it is incomparable with linear and with context-free languages.

The membrane systems class was shown to be closed under union, permutations, erasing/renaming morphisms. It is not closed under intersection, intersection with REG, complement, concatenation or the mirror image.

Some questions are still not answered, like inclusion in $ET0L$ and MAT, sharper lower and upper bounds and closure under arbitrary morphisms.

References

[1] A. Alhazov, D. Sburlan: Static Sorting P Systems. *Applications of Membrane Computing* (G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, Eds.), Natural Computing Series, Springer-Verlag, 2005, 215–252.

[2] F. Bernardini, M. Gheorghe: Language Generating by means of P Systems with Active Membranes. *Brainstorming Week on Membrane Computing*, Technical Report 26, Rovira i Virgili University, Tarragona, 2003, 46–60.

[3] Gh. Păun: *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.

[4] Gh. Păun, G. Rozenberg, A. Salomaa, Eds.: *Handbook of Membrane Computing*. Oxford University Press, 2009.

[5] G. Rozenberg, A. Salomaa: *Handbook of Formal Languages*, vol. 1-3, Springer, 1997.

[6] P systems webpage. `http://ppage.psystems.eu/`